UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

MASTER COPY - FOR REPRODUCTION PURPOSES

(2)

OCUMENTATION PAGE

AD-A205 630

1a. REPORT : Un	1b. RESTRICTIVE MARKINGS
2a. SECURITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSI ... SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARO 25960.1-EL-SBI
6a. NAME OF PERFORMING ORGANIZATION Atlantic Aerospace Electronics Corporation	6b. OFFICE SYMBOL (If applicable)
6c. ADDRESS (City, State, and ZIP Code) Waltham, MA 02154	7a. NAME OF MONITORING ORGANIZATION U. S. Army Research Office
	7b. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U. S. Army Research Office	8b. OFFICE SYMBOL (If applicable)
8c. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAL03-88-C-0018
	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Hardware-Specific Symbolic Signal Manipulation

12. PERSONAL AUTHOR(S)
Cory Myers

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 8/15/88 TO 2/14/89	March 1989	47

16. SUPPLEMENTARY NOTATION
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Symbolic Signal Manipulation, Signal Representations, Computer Aided Design, Digital Signal Processing, Signal Processing Algorithms

19. ABSTRACT

The goal of this Phase I SBIR was to demonstrate extensions to current systems for signal representation and manipulation. The long term goal of this SBIR is to develop CAD assistants for the design, development, and test of signal processing algorithms. In Phase I we did the following:

- Surveyed several current DSP processors and developed an understanding of the common features of these processors.
- Developed an explicit representation for an abstract DSP processor that can be specialized to model specific DSP processors.

(Over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

- Developed software for resource usage measurement on the abstract DSP processor.
- Interfaced the processor representation with E-SPLICE, a system for the symbolic representation and manipulation of signals, thus demonstrating hardware-specific symbolic signal manipulation.
- Rapidly extended the signal representation facilities to incorporate descriptions of multirate filter bank structures, a signal processing structure of current interest to the Army.
- Examined the area of direction finding and emitter location to determine the applicability of the signal representation and manipulation techniques to this area.
- Determined that the area of direction finding and emitter location, an important area to the Army, is an area that would benefit from application of the signal representation and manipulation techniques used in Phase I.

ARO 25960.1-GL-SBI

ATLANTIC AEROSPACE ELECTRONICS CORPORATION
470 Totten Pond Road • Waltham, MA 02154 • Phone (617) 890-4200 • FAX (617) 890-0224

Hardware-Specific Symbolic Signal Manipulation

SBIR Phase I Program Final Report

March 13, 1989

Prepared for
Army Research Office
under
Contract No. DAAL03-88-C-0018

Table of Contents

Figures

1. Project Summary

The goal of this SBIR is to develop the fundamental components of a computer aided design (CAD) assistant for the design and implementation of signal processing algorithms of importance to the Army. Current tools for the design and analysis of signal processing algorithms are neither sufficiently powerful nor sufficiently specialized to provide high leverage towards the development of efficient signal processing algorithms for important problem areas, such as channelization or direction finding. Phase I of this work investigated techniques for overcoming current limitations. In Phase II we propose to develop and deliver a complete system for an important signal processing area - direction finding and emitter location. In Phase III we would extend the system developed in Phase II to other problem areas and would market the resulting systems.

Recent developments in the area of "knowledge-based signal processing" have led to the development of object-oriented approaches to signal representation and manipulation. These techniques have been demonstrated to drastically reduce the amount of time required to design, develop, and test signal processing algorithms. These techniques have also been demonstrated to provide facilities for the automatic analysis and rearrangement of signal processing algorithms in an attempt to automatically generate efficient signal processing algorithm implementations. However, these techniques do not provide the tools needed by the Army in its signal processing algorithm development. Specifically, these techniques are not designed with specific, important problem areas in mind and these techniques use computational models that are overly restrictive.

The Phase I SBIR work investigated the possibility of overcoming the restrictions of current signal processing algorithm design systems. Specifically, we investigated the ability to extend current symbolic signal manipulation systems to incorporate more realistic models of the hardware on which signal processing algorithms would be run and to work on problems in a signal processing area of interest to the Army. We investigated single-chip digital signal processing (DSP) processors. We found that much of the common structure of these processors could be captured in a software model and we developed an explicit representation of an abstract processor that incorporates many of the common features of the DSP processors we investigated. We discussed signal processing problem areas with representatives from the Army and chose the area of multirate filter banks as a problem area of interest. We extended a current symbolic signal manipulation system to represent signal processing problems from this area and we interfaced the signal representation with the processor representation. In addition, we identified another signal processing area of importance to the Army - direction finding and emitter location - in which we propose, as the Phase II effort, to develop and deliver a CAD system for algorithm design, development, and test.

2. Background

The process of developing and implementing complicated signal processing algorithms in efficient forms is a difficult, time-consuming, and error-prone process. The cost of developing an efficient implementation of a particular signal processing algorithm is often a major cost in a system. Methodologies and tools to aid in the development of signal processing implementations are not generally available. These problems are particularly critical when large parallel systems are required for the implementation of the signal processing algorithm.

One of the most significant recent trends in signal processing has been the growing interest in "knowledge-based signal processing" (KBSP), the application of symbolic knowledge representation and reasoning techniques to the design of signal processing systems [1, 2, 3, 4, 5]. A major motivation for the study of KBSP is the need for computer-based tools to handle the increasing complexity of the signal processing algorithms which are being applied to practical problems.

Work in the area of KBSP has led to two significant developments. The first is the application of software development tools and environments taken from the area of symbolic processing applied to the area of signal processing. The application of these tools and environments has led to the creation of several powerful, interactive, signal processing environments, such as the Signal Representation Language (SRL) [6], the Integrated Signal Processing System (ISP) [7], and the Signal Processing Language and Interactive Computing Environment (SPLICE) [8]. Work with these systems has shown that they can be used to shorten the time required for signal processing algorithm development by factors of from two to five.

The second significant development in KBSP is the concept of "symbolic signal analysis and manipulation" [8, 9]. Symbolic signal analysis and manipulation is the explicit representation and manipulation of the *form* of signal processing algorithms and the automatic determination of properties of these algorithms. This is in contrast to standard signal processing in which the *numerical* values of signals are manipulated.

The symbolic analysis of signals involves determining the properties of signals from the expressions that define them rather than by examination of the numerical values of the signals. For example, the non-zero support of the convolution of $x[n]$ and $h[n]$, $y[n] = x[n] * h[n]$, can be determined from the non-zero supports of the signals $x[n]$ and $h[n]$ and properties of the convolution operator. Similarly, the number of multiplications required to compute one sample of $y[n]$ in a direct-form implementation can be determined from information about the symmetries and non-zero supports of $x[n]$ and $h[n]$ without any computation of signal values.

The transformation of signal processing expressions to determine equivalent ways of expressing the same computation is similar to the rearrangement of algebraic expressions done by systems such as MACSYMA [10]. For example, the time

domain convolution $y[n] = x[n] * h[n]$ can be rearranged to its frequency domain version, i.e. $y[n] = FT^{-1}\{FT\{x[n]\} \cdot FT\{h[n]\}\}$. Transformations may be used to simplify expressions for presentation, to assist in signal property analysis by changing the form of the expression to one in which the desired property can be more easily computed, and to search for low cost implementations.

E-SPLICE [8, 9], a rule-based system for the symbolic manipulation of signal processing algorithms, has demonstrated the ability to automatically analyze and transform signal processing algorithms. As an example of the use and power of symbolic manipulation of signals, E-SPLICE has been used in the analysis and transformation of multirate signal flow diagrams, particularly in the search for efficient implementations of multirate networks. For a particular multirate network, this search involved the automatic analysis of the computational costs of each implementation of the network and the automatic transformation of one implementation into another in order to search for a good implementation. In at least one case this automatic search process was able to find a *new* implementation of a particular multirate network.

The combination of interactive signal processing environments with symbolic signal analysis and manipulation point the way to the development of a computer aided design (CAD) assistant for the development and implementation of signal processing algorithms. Such an assistant could potentially reduce the costs of signal processing work and could also, in some cases, find algorithm implementations which are superior, in terms of problem-dependent cost criteria, to those achieved without it. One mechanism for such improvement is that a design assistant allows the user to perform algorithm optimizations over a larger space of candidate implementations. In addition, a system which can autonomously generate and evaluate alternative implementations of an algorithm may discover a superior implementation which a human designer would overlook because it differs from the kind of implementation normally used. The case in which E-SPLICE found a new implementation is an example of this.

3. Research Program

It is the goal of this SBIR research to develop the fundamental components of a computer aided design (CAD) assistant for the design and implementation of signal processing algorithms of importance to the Army. To achieve this goal we need both to enhance the tools for the analysis of signal processing algorithms and to demonstrate the utility of these tools in a problem area of interest to the Army. Although SRL, ISP, SPLICE, and E-SPLICE demonstrated the potential of signal representation, symbolic signal analysis, and symbolic signal manipulation and thus provide some of the most fundamental pieces for a CAD assistant for signal processing, they certainly do not provide a usable CAD system. Shortcomings of these systems include the following:

- These systems are not designed with specific, important problem areas in mind. For example, SPLICE has been used to study high-resolution spectral estimation algorithms but does not contain all the tools required to completely analyze an emitter location problem.

- These systems are not available on general-purpose workstations. They are designed for research environments and run on special-purpose, i.e. Lisp, workstations.

- The models these system use for signal analysis are overly restrictive. For instance, E-SPLICE uses general-purpose symbolic analysis and manipulation methods that do not maintain realistic hardware models. As such, the search for efficient implementations does not necessarily yield solutions that are optimal for a particular hardware architecture.

Overcoming these problems would allow the development of specialized CAD assistants for the design and implementation of signal processing algorithms. Such assistants would have specialized knowledge of particular problem domains and would help in the development, testing, and realization of signal processing algorithms. It would provide the tools to significantly reduce the time required for a person to design, test, refine, and implement an algorithm.

3.1. Phase I Objectives

The Phase I technical objective was to demonstrate that one of the limitations of current CAD systems for signal processing could be overcome. Specifically, the goal of Phase I was to demonstrate the viability of hardware-specific signal analysis and manipulation for the design of efficient signal processing implementations. This work involved three objectives:

1. Development of an explicit representation for a particular hardware architecture.

2. Extension of existing symbolic analysis and manipulation techniques to use the architectural representation.

3. Demonstration of hardware-specific symbolic signal analysis and manipulation using the developed hardware representation and symbolic manipulation techniques.

3.2. Phase II and Phase III Objectives

Phase II work will be directed towards a more complete design and development of a specialized CAD system for signal processing algorithm design and analysis. Towards this goal Atlantic Aerospace has interacted closely with the Army Signals Warfare Center and has determined that the problem of direction finding and emitter location is a specialized signal processing problem area of importance to the Army that could benefit from more extensive development of the techniques demonstrated in Phase I. As such, Atlantic Aerospace has proposed the development and delivery of a direction finding and emitter location algorithm testbed as the Phase II program. More on the topic of direction finding and emitter location and its relationship to signal representation techniques can be found in Section 7 and in the Phase II proposal [11].

Phase III work will be directed towards generalizing and extending the direction finding and emitter location testbed developed in Phase II towards other problem areas of interest to the Army. This work would result in a line of signal processing design and analysis testbeds that Atlantic Aerospace would produce as commercial products.

4. Phase I Technical Approach

The Phase I work was directly supervised by Army Signals Warfare Center who provided guidance on signal processing problem areas and architectures of interest. The technical objectives of the Phase I program were accomplished in a set of six tasks, as follows:

1. Examination, by Atlantic Aerospace, of current signal processing algorithms and architectures. Signal processing problems and architectures were discussed with the Army Signals Warfare Center to understand the places in which signal representation and manipulation could provide the highest leverage. Researchers at MIT currently interested in signal processing architectures were consulted on a regular basis also.

2. Selection, in conjunction with the Army Signals Warfare Center, of the class of hardware architectures consisting of single-chip DSP microprocessors as the architecture of interest.

3. Implementation, by Atlantic Aerospace, of software for the explicit representation of an abstract single-chip DSP processor.

4. Selection, in conjunction with the Army Signals Warfare Center, of the class of signal processing problems consisting of multirate filter banks as the problem area of interest.

5. Extension of the E-SPLICE software, by Atlantic Aerospace, to include the representation and manipulation of multirate filter banks and to interface with the processor representation.

6. Demonstration and evaluation of the utility of hardware-specific symbolic signal manipulation for the signal processing architecture and problem chosen.

In addition to the previous six tasks Atlantic Aerospace also, at the request of the Army Signals Warfare Center, examined the class of signal processing problems involving direction finding and emitter location to determine if the techniques developed in Phase I would be applicable to that problem area. This work helped to shape the Phase II proposal.

5. Processor Survey

An important portion of the Phase I program was involved in a survey of available single-chip DSP microprocessors. In early discussions with the Army Signals Warfare Center, it became clear that a signal processing architecture of immediate interest is the class of single processor DSP cards containing such DSP processors as the TMS 32020 and the AT&T DSP 32. It was felt that an intensive examination of the range of DSP processing cards was beyond the scope this Phase I effort. We therefore restricted our investigations to the single-chip DSP processors without examination of the systems in which they are embedded. We examined the following single-chip DSP processors:

1. The Texas Instruments TMS 32010.
2. The Texas Instruments TMS 32020.
3. The Texas Instruments TMS 32030.
4. The AT&T DSP 32.
5. The Motorola DSP 56000.

Examination of these processors concentrated on identification of the key architectural properties of these processors and the identification of commonality among these processors. Topics of interest in surveying the processors included the following:

1. Number and size of on-chip memory.
2. Number and size of off-chip memory.
3. Instruction cycle time.
4. Number of cycles for most common operations, e.g. addition, multiplication, multiply-accumulate.
5. Overhead of looping.
6. Bandwidth between processor, internal memory, and external memory.
7. Hardware word size.
8. Fixed vs floating point representation.
9. Input/Output bandwidth
10. Special addressing modes.

A summary of the information that was collected about these processors is given in the processor outlines of Appendix A. An excellent comparative description of several DSP processors is given in [12].

Our survey of single-chip DSP processors led us to the following conclusions:

- All the single-chip DSP processors studied rely on a memory hierarchy. This hierarchy includes special-purpose registers, such as the accumulator in the AT&T DSP 32, internal memory, and external memory.

- All the single-chip DSP processors studied use special-purpose hardware and data paths to achieve speed. For example, the output of the floating point multiplier in the AT&T DSP 32 leads directly to one of the input ports of the floating point adder. This structure makes it very easy to pipeline a sequence of multiply-accumulates, the common operation in FIR filtering. However, a single multiply requires extra time to move the result through the adder and back out again.

- The processing rate of all the single-chip DSP processors studied is limited by data movement rather than by the processing time of the functional units. These processors are designed with internal memory and registers whose speed is matched to the functional units, i.e. if the data is in the right places then the functional units almost never wait for the memory and the memory almost never waits for outputs from the functional units. Most processing delays are due to the data not being in the optimal location.

- Special programming techniques are key to getting the maximum speed out of these processors. For example, an FFT program for the Motorola DSP 56000 should make use of its ability to perform bit-reversed addressing. It is difficult to build a compiler that can recognize all the important programming techniques and, hence, having a library of optimized subroutines is important.

6. Hardware-Specific Symbolic Signal Manipulation

Software for hardware-specific symbolic signal manipulation was developed and demonstrated in two tasks:

1. Development of an explicit representation for DSP processors. This development was based on the processor survey and resulted in the development of an abstract processor representation and development of a computational resource usage model for the abstract processor. This processor representation was interfaced to E-SPLICE for the purpose of hardware-specific symbolic signal manipulation.

2. Development of software for the representation and manipulation of multirate filter banks. This work resulted in extensions to the SPLICE and E-SPLICE software packages. These extensions provide tools for the design, test, analysis, and manipulation of multirate filter banks.

6.1. Processor Representation

The processor survey provided the material for the development of an abstract DSP processor model. Because of the basic similarities of the surveyed DSP processors it was possible to design a representation of an abstract processor and to then specialize this abstract processor to model a specific DSP processor. The following design decisions were made in the development of the processor representation:

- An abstract processor model would be developed that could be specialized to provide a model for any of the DSP processors we examined.

- Modeling the computation performed by the surveyed DSP processors at a fine-grain level, e.g. modeling an inner-product as a sequence of individual multiplies and adds or even as a sequence of individual multiply-accumulates, ignores the speed that can be achieved by pipelining. The representation we developed can directly model high-level operations, such as inner product or vector add, and only uses fine-grain analysis when no high-level operation is available.

- The processor representation includes explicit representation of memory. The importance of keeping track of where data is stored and how easily it can be accessed was clear from the survey.

- The processor representation does not provide the "optimal" representation for any given DSP processor because the representation does not make use of all the available optimizations of any given

processor. The "optimal" representation of any given DSP processor would require the use of a compiler for that processor and analysis using an emulator.

We designed, developed, and tested a representation for an abstract DSP processor. This processor was used in our experiments on symbolic manipulation. We also modeled the TMS 32010 by specializing the abstract DSP processor.

The processor representation developed used an object-oriented approach, i.e. a processor to be modeled was represented as a data structure with associated operations, such as "add vectors". Different processor types could be used simultaneously because the user of a processor object did not need to know the type of the processor object, only that the processor object would respond to certain messages. The processor representation consists of approximately 3000 lines of Common Lisp with Flavors extensions running on a Symbolics Lisp Machine.

The abstract processor representation consists of functional units, such as adders and multipliers, and three levels of memory - registers, internal RAM, and external RAM. The functional units are all two-input one-output units. Processing time for basic operations is defined by a table that takes as inputs the requested operation, i.e. add or multiply, the location of the two operands, i.e. register, internal, or external memory, the destination of the output, and a flag specifying if the operation is occurring within a loop or outside of a loop. The table produces as output the number of cycles required for the operation given the input and output locations and the looping context. A typical entry for the generic processor is as follows:

Operation: Multiply
Input A Location: Internal Memory
Input B Location: Register
Output Destination: Register
Loop Mode: Non-Looping
Cycles: 1

This entry specifies that one cycle is required for a multiply when one input is in internal memory and the other is in a register and the output is to be written to a register. Specific DSP processors are created by specializing the memory sizes and the cycle times in the table. For example, the TMS 32010 is modeled using a single internal memory of size 288 bytes.

Hardware-specific symbolic signal manipulation is accomplished by combining the processor representation with the signal representation and manipulation techniques of E-SPLICE. The relationship between E-SPLICE and a processor object is illustrated in Figure 1. E-SPLICE makes a request for evaluation of computational resources for some operation, such as "inner product". The processor object takes as input the requested operation, a list of input operands, and the output location. The processor object returns a resource usage description. This description provides

information on how many cycles of time were used for this operation, how much memory was used, and how much i/o had to be performed. To compute the resource usage the processor performs the following steps:

1. Data movement. The list of input operands is interpreted as a "request" for operand locations, it is not interpreted as a requirement. If the request can not be satisfied then the processor object is free to move the data around. For example, we may request a TMS 32010 processor to multiply two 1000-point vectors from internal memory but the processor will realize that it does not have that much internal memory and will move the operands to external memory. Not requiring the user of a processor object to know about the sizes of memory simplifies the usage of the processor representation.

2. Computational cycle time analysis. The processor object uses the operand locations resulting from step 1 and the table of cycle times to determine the number of cycles required for computation.

3. Output resource usage. The processor object returns a description of the use of each different memory, the amount of i/o, the number of cycles, and a description of how much time is spent in loop overhead.

By far the largest fraction of the processor computational analysis code is devoted to the data movement portion. This reflects the observation in the processor survey that a key factor in the computational performance of the single-chip DSP processors is the data movement. An example of the results of asking a generic processor with 288 bytes of internal memory for the computational costs of performing a 30-point vector inner product is given in Figure 2. This shows that a 30-point inner product uses 120 cycles of processing plus 90 cycles of loop overhead. The operation requires no i/o and no external memory. Up to 122 bytes (two bytes per point times 30 points times two vectors plus two bytes for the result) of internal memory are used for the operation. Figure 3 shows the result of a 120-point vector inner product on a generic processor with 288 bytes of internal memory. This shows that the number of cycles has increased with respect to a 30-point vector but, more importantly, the processor has determined that it must perform some i/o to do this operation. The figure shows that the processor requires 96 i/o operations (all but 96 of the input values can fit in the internal memory) to perform this computation.

Implemented in our generic processor model are the fundamental arithmetic operations, the point-wise vector operations of multiply and add, and the inner product operation. Vector operations were defined for general N vector inputs, i.e. the inner product operation could perform $\sum x[n]y[n]$, $\sum x[n]y[n]z[n]$, etc. The processor object responds to requests for operations for which it does not have a description, such as "Filter", with a "I do not understand, please decompose". It is the responsibility of the E-SPLICE software to decompose the operation into smaller pieces, ask the processor object for computational analysis of the smaller pieces, and

to collect the results. The choice of decomposition is up to E-SPLICE. For example, for an FFT E-SPLICE would decompose the problem into a sequence of vector multiplies and vector adds. E-SPLICE decomposes a 4-point FFT as shown in Figure 4. This shows E-SPLICE breaking the input into pieces, performing vector operations on the pieces, defining temporary variables, and writing the results back. Important things to note about this decomposition are the following:

- This decomposition reuses temporary variables when they are not required. This helps to reduce the amount of data movement required. The particular way in which temporary variables are reused, however, is written into E-SPLICE. A better system would note that some variables are no longer needed and would reuse them automatically.

- E-SPLICE uses the radix-2 FFT. E-SPLICE could easily be extended to use radix-4.

- E-SPLICE assumes that the twiddle factors for the FFT already exist.

- E-SPLICE's decomposition could not take advantage of a processor with bit-reversed indexing, such as the Motorola DSP 56000. A processor object modeling this processor should contain within it an analysis routine for the computation costs of the FFT rather than relying on E-SPLICE to form the decomposition.

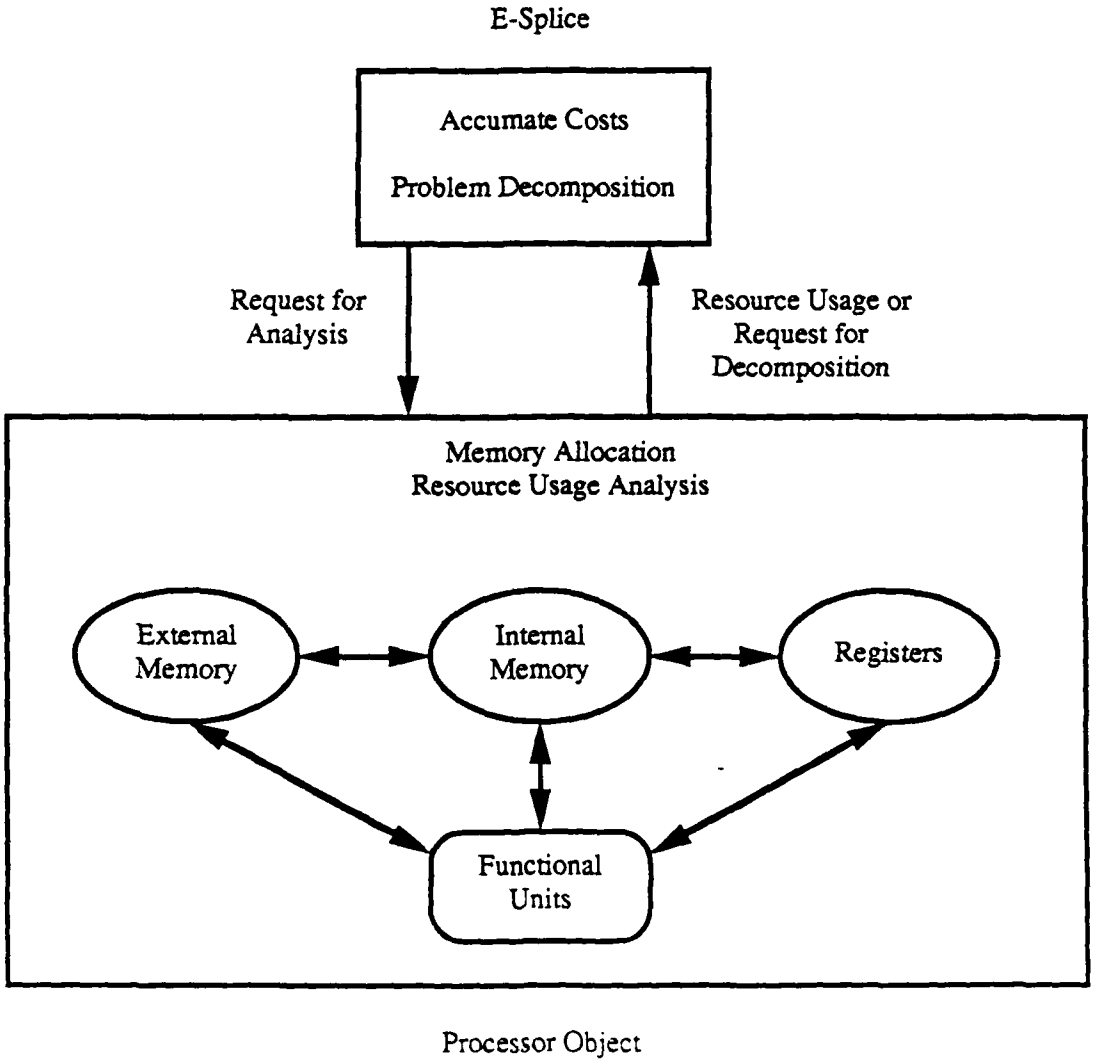


Figure 1. Relationship between E-SPLICE and a processor object.

```
Vector Inner Product of Size 30 ==>

resources of #<GENERIC-PROCESSOR 351472451>
#<RESOURCE-USE 54444721>, an object of flavor RESOURCE-USE,
 has instance variable values:
  CYCLES:  120
  LOOP-CYCLES:  90
  MEMORY-USAGE:  (#<MEMORY 54444672> #<MEMORY 54444666>)
  IO-USAGE:  NIL
  PROC-UNIT-USAGE:  NIL
#<MEMORY 54444672>, an object of flavor MEMORY,
 has instance variable values:
  LOCATION:  EXTERNAL-RAM
  SIZE:  0
#<MEMORY 54444666>, an object of flavor MEMORY,
 has instance variable values:
  LOCATION:  INTERNAL-RAM
  SIZE:  122
```

Figure 2. Output from a computational analysis of a 30-point vector inner product on the generic processor.

```
Vector Inner Product of Size 120 ==>

resources of #<GENERIC-PROCESSOR 351472451>
#<RESOURCE-USE 54445155>, an object of flavor RESOURCE-USE,
 has instance variable values:
  CYCLES:  576
  LOOP-CYCLES:  360
  MEMORY-USAGE:  (#<MEMORY 54445065> #<MEMORY 54444770>)
  IO-USAGE:  #<IO-CHANNEL 54445125>
  PROC-UNIT-USAGE:  NIL
#<MEMORY 54445065>, an object of flavor MEMORY,
 has instance variable values:
  LOCATION:  EXTERNAL-RAM
  SIZE:  0
#<MEMORY 54444770>, an object of flavor MEMORY,
 has instance variable values:
  LOCATION:  INTERNAL-RAM
   SIZE:  288
#<IO-CHANNEL 54445125>, an object of flavor IO-CHANNEL,
 has instance variable values:
  MIN-USAGE:  1.0
  MAX-USAGE:  1
  TOTAL-USAGE:  96
  TYPE:  SOLE-IO
```

Figure 3. Output from a computational analysis of a 120-point vector inner product on the generic processor.

```
(SHOW-DECOMPOSITION (FFT X-REAL X-IMAG 4)) ===>

Vector add (PART X-REAL 0 1) by (PART X-REAL 2 3) to TEMP-1
Vector add (PART X-IMAG 0 1) by (PART X-IMAG 2 3) to TEMP-2
Vector sub (PART X-REAL 0 1) by (PART X-REAL 2 3) to TEMP-3
Vector sub (PART X-IMAG 0 1) by (PART X-IMAG 2 3) to TEMP-4
Vector mul TEMP-3 by TWIDDLES-4-1-REAL to TEMP-5
Vector mul TEMP-3 by TWIDDLES-4-1-IMAG to TEMP-6
Vector mul TEMP-4 by TWIDDLES-4-1-REAL to TEMP-7
Vector mul TEMP-4 by TWIDDLES-4-1-IMAG to TEMP-8
Vector sub TEMP-5 by TEMP-8 to TEMP-3
Vector add TEMP-6 by TEMP-7 to TEMP-4
Vector add (PART TEMP-1 0 0) by (PART TEMP-1 1 1) to
       (PART X-REAL 0 0)
Vector add (PART TEMP-2 0 0) by (PART TEMP-2 1 1) to
       (PART X-IMAG 0 0)
Vector sub (PART TEMP-1 0 0) by (PART TEMP-1 1 1) to
       (PART X-REAL 2 2)
Vector sub (PART TEMP-2 0 0) by (PART TEMP-2 1 1) to
       (PART X-IMAG 2 2)
Vector add (PART TEMP-3 0 0) by (PART TEMP-3 1 1) to
       (PART X-REAL 1 1)
Vector add (PART TEMP-4 0 0) by (PART TEMP-4 1 1) to
       (PART X-IMAG 1 1)
Vector sub (PART TEMP-3 0 0) by (PART TEMP-3 1 1) to
       (PART X-REAL 3 3)
Vector sub (PART TEMP-4 0 0) by (PART TEMP-4 1 1) to
       (PART X-IMAG 3 3)
```

Figure 4. Decomposition of a 4-point FFT by E-SPLICE into smaller operations.

6.2. Multirate Filter Bank Representation and Manipulation

In addition to the development of a representation for DSP processors, software was also developed for the representation and manipulation of multirate filter bank systems. Multirate filter bank systems are an important component in many signal intercept and analysis problems. An excellent description of the signal processing properties of multirate systems is given in [13] and this, as well as [14, 15, 16, 17], were used as guides to the development of the software. This software was based on existing tools in SPLICE [8]. The resulting software includes the following signal processing capabilities:

- Modulation of signals by either real sinusoids or complex exponentials.
- Convolution of signals.
- Design of FIR filters using minimax approximation techniques.
- Upsampling and downsampling of signals.
- The DFT and the FFT.
- Polyphase filter structures for filter-downsample and upsample-filter.
- Single channel upsample, filter, and modulate for one channel of a filter bank modulator.
- Single channel demodulate, filter, and downsample for one channel of a filter bank demodulator.
- Polyphase filter bank synthesizer.
- Polyphase filter bank demodulator.

An example of a portion of software developed for this SBIR is as follows:

```
(DEFINE-COMPOSITION FILTER-BANK-SYNTHESIS-CHANNEL
                    (INPUT FREQUENCY FILTER UPSAMPLE-RATE)
  "Single channel of a filter bank synthesizer"
  (SEQ-MULTIPLY
    (SEQ-CONVOLVE
      (SEQ-UPSAMPLE INPUT UPSAMPLE-RATE)
      FILTER)
    (COMPLEX-EXPONENTIAL-SEQ FREQUENCY)))
```

This shows the definition of a single channel of a filter bank synthesizer. The input signal `INPUT` is first upsampled by the upsampling rate `UPSAMPLE-RATE`, then is filtered

by FILTER, and the result is modulated by a complex exponential of frequency FREQUENCY. This definition shows the close resemblance between the software for the operation and the mathematical description of the operation $y[n] = (upsample(x[n], M) * h[n]) \cdot e^{j\omega n}$. The naturalness of this code is a result of the object-oriented signal representation used in SPLICE. This representation allows the user to hide many of the details of the internal signal representation from those operations that do not require them. As a result, less than a week of the Phase I effort was devoted to extending the SPLICE software to incorporate all the operations required for multirate filter bank signal processing.

An example of signal processing for filter banks using the software developed in this program is given in Figure 5. This figure shows ten plots. The leftmost column of three plots shows three input bit streams. Nine bits are shown in each input signal. The next column shows the single-channel filter bank outputs. Each of the bit streams is encoded using a three-bit QAM coder, upsampled, filtered with a rectangular window impulse response, and modulated using a different carrier for each channel. The resulting three single-channel filter bank outputs are summed and the result is shown in the middle plot. The next column shows three channels of a filter bank analysis system. The middle signal is demodulated and filtered. The three plots show that the result of demodulating and filtering are similar to, but not identical with, the individual synthesis channels. This is because the rectangular window that is used for synthesis and analysis is not an ideal low-pass filter and thus each channel exhibits some leakage from nearby channels. However, the last column of three plots shows that QAM decoding of the filter bank analysis channels results in the correct bit streams.

Figure 6 shows another example of the the use of the signal processing software for the design of a particular multirate filter bank. In this particular problem we wish to design a pulse-shaping filter whose bandwidth is approximately 7 Hz for a signal that has been sampled at 4000 Hz. Three example pulse shaping filter impulse responses are shown in the top row - a rectangular window, a Hamming window, and the convolution of a Hamming window with itself. The next row shows the log magnitude of the frequency response of each of these pulse shaping filters and shows, for these filter impulse responses, the increasing out-of-band rejection as the impulse response length increases.

Software for the symbolic analysis and manipulation of multirate filter bank systems was developed using the rule-based capabilities of E-SPLICE for symbolic signal manipulation. Rules for the following signal manipulations are used in the analysis and manipulation of multirate systems:

- Duality between time-domain convolution and frequency domain multiplication.
- Ability to simplify FIR convolutions for symmetric impulse responses.

- Rules for combinations of upsamples, downsamples, and shifting operations. For example, upsampling by a factor of M immediately followed by downsampling by a factor of $L \cdot M$ is equivalent to downsampling by a factor of L.

- Rules for the single-channel polyphase filter structures for filtering followed by downsampling and upsampling followed by filtering.

- Rules for the multi-channel polyphase filter bank analysis and synthesis structures.

An example of a rule used for the analysis of multirate systems is as follows:

```
(DEFINE-DOWNSAMPLE-SIMPLIFICATION-RULE DOWNSAMPLE-OF-DOWNSAMPLE
   "Downsampled downsampled signal is a downsampled signal"
   (:FORM
     (DOWNSAMPLE (DOWNSAMPLE ?SIG ?L1) ?L2))
   (:RESULT
     (DOWNSAMPLE SIG (* L1 L2))))
```

The rule is made up of a `:FORM` part which specifies under what conditions the rule applies and a `:RESULT` part which specifies what the result of running the rule is. The expressions `?SIG`, `?L1`, and `?L2` are variables in the rule and will match the appropriate values when the rule is applied. This rule states that if a signal is downsampled by L_1 and the result of that is downsampled by L_2 then the result can be simplified to a single downsample of the original signal by a factor $L_1 \cdot L_2$. This rule is applied by the E-SPLICE system whenever it is attempting to simply expressions and determines that the expression to be simplified is of the form of a downsample of a downsample.

An important aspect of the development of rules for the analysis of multirate filter banks was the choice of representation for the filter banks. An analysis filter bank of N channels could be represented by a collection of N expressions, one for each channel. Such a representation is by far the most flexible but has a severe drawback when used with an automatic manipulation system. The drawback is that certain problems that one uses an automatic manipulation system for cause a large number of intermediate (or final) expressions to be generated. This issue is known in the symbolic manipulation community as "intermediate expression swell". For example, a simple single-channel convolution followed by a downsampling operation can be rearranged into over a hundred different expressions [8] - time-domain convolution followed by downsampling, frequency-domain convolution followed by downsampling, polyphase filter structure containing time-domain convolution, polyphase filter structure containing frequency-domain convolution, etc. In the straightforward analysis of a multirate filter bank system with N channels such a growth of expressions would be repeated for each of the N channels causing an exponential growth that would make all but the smallest of problems intractable. Methods for restricting this growth while still performing a reasonable search for interesting signal

processing rearrangements is an area of active research [18]. Rather than attempting to solve this problem in the course of this SBIR we chose instead to develop a less flexible representation for multi-channel filter banks. Specifically, we represented a multi-channel filter bank analysis system not as a collection of N different channels but rather as an aggregated system with a single channel whose output, at each instant in time, was a set of N numbers. This choice of representation had the following implications:

- The threat of exponential growth in the number of expressions was avoided.
- Although the rules existed for the manipulation of individual channels, it was necessary to add rules for the manipulation of the aggregate system.
- The number of rules for the manipulation of the aggregate system was small; specifically, the two rules for polyphase filter banks - both analysis and synthesis filter banks - were the only rules that manipulated the aggregate structure.
- The small number of rules for aggregate systems implied rearrangement and simplification problems involving the aggregate structures generated very few choices.

These factors illustrate a general idea - the use of higher level structures reduces the search space but may result in a system that is not as thorough as it should be. If the symbolic manipulation system were to explore the rearrangement of individual channels separately it might find a better solution by doing different things to different channels.

An example of the manipulation of the aggregate multirate filter-bank representation is given in Figure 7. This shows the user requesting all the rearrangements of a multirate filter bank analysis system. The user specifies that the analysis system takes as input the signal $x[n]$and each channel of the analysis system uses a filter with impulse response $h[n]$. The user specifies that the filter bank analysis system is to have 32 output channels each of which is to be generated by modulating $x[n]$, filtering that by $h[n]$, and downsampling the result by 32. The system responds with methods for performing this operation. The first - `(PARALLEL-CHANNELS MODULATE 32 (DOWNSAMPLE (CONVOLVE (MODULATE X) H) 32))` - is the expression for 32 parallel modulation channels each of which consists of modulating $x[n]$, convolving the result in the time-domain with $h[n]$, and downsampling the result by 32. The second - `(PARALLEL-CHANNELS MODULATE 32 (DOWNSAMPLE (IFT (MULTIPLY (FT (MODULATE X)) (FT H))) 32))` - is the expression for 32 parallel modulation channels each of which consists of modulating $x[n]$, convolving the result in the frequency-domain with $h[n]$, and downsampling the result by 32. The third expression also is for 32 parallel channels using frequency-domain convolution but the

parallelism is over shifts of the Fourier Transform of $x[n]$ (corresponding to modulation). The fourth expression is for 32 parallel modulation channels but the filter and downsample operations are replaced by polyphase filter and downsample. The fifth expression - `<POLYPHASE-FILTER-BANK-ANALYSIS X H 32>` - is an aggregate representation for the polyphase critically sampled analysis filter bank.

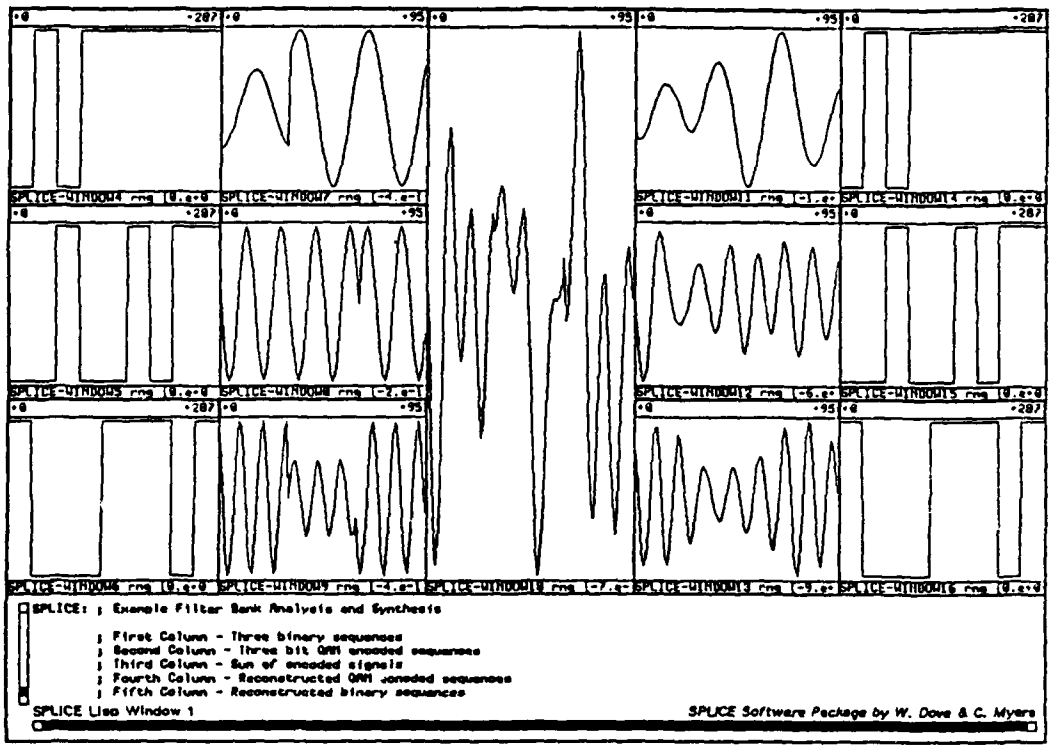


Figure 5. Example signal processing for filter banks using software developed for this program.

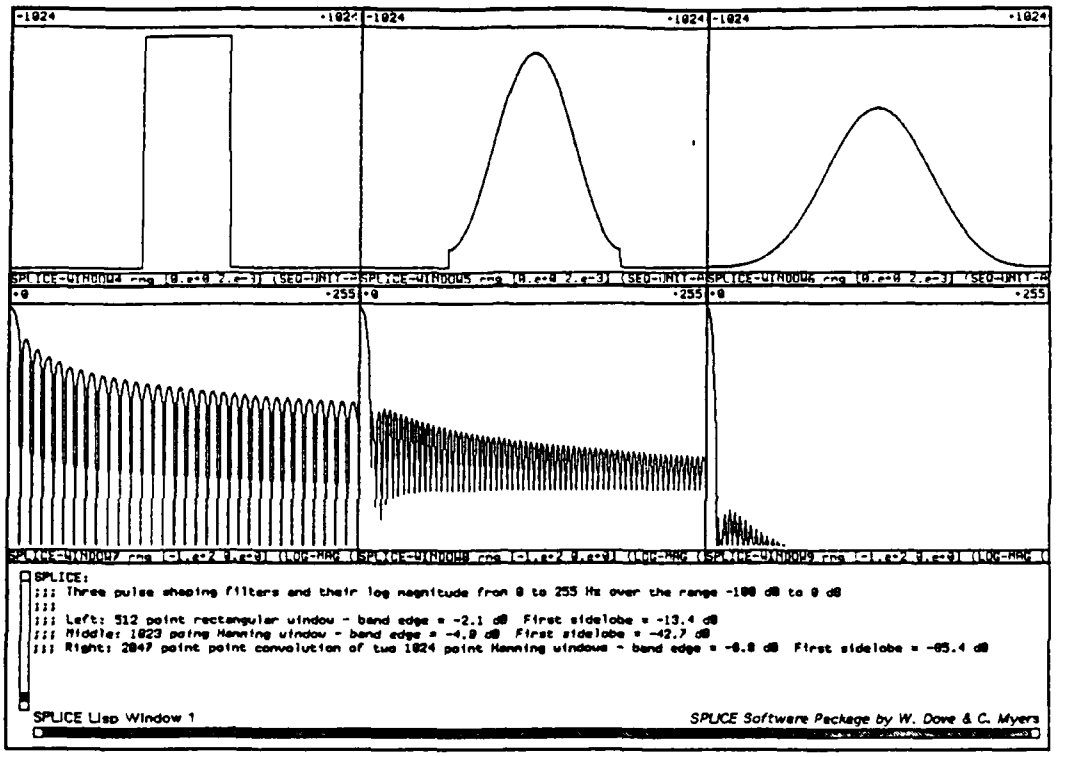


Figure 6. Example pulse shaping filter design.

```
E-SPLICE: (ALL-REARRANGEMENTS
            (ANALYSIS-FILTER-BANK
              :INPUT X :FILTER H
              :CHANNELS 32 :DOWNSAMPLE 32))

====> ((PARALLEL-CHANNELS MODULATE 32
         (DOWNSAMPLE (CONVOLVE (MODULATE X) H) 32))

       (PARALLEL-CHANNELS MODULATE 32
         (DOWNSAMPLE (IFT (MULTIPLY (FT (MODULATE X)) (FT H)))
                     32))

       (PARALLEL-CHANNELS SHIFT 32
         (DOWNSAMPLE (IFT (MULTIPLY (SHIFT (FT X)) (FT H)))
                     32))

       (PARALLEL-CHANNELS MODULATE 32
         (POLYPHASE-FILTER-AND-DOWNSAMPLE (MODULATE X) H 32))

       (POLYPHASE-FILTER-BANK-ANALYSIS X H 32) ...)
```

Figure 7. Manipulation of a multirate filter bank structure.

7. Direction Finding and Emitter Location

The main thrust of the work in this program is the signal processing of multirate filter banks. At the request of the Army Signals Warfare Center, we also examined the area of direction finding and emitter location. This is an area in which Atlantic Aerospace personnel have much experience [19, 20, 21, 22]. The goal of this investigation was to determine if the area of direction finding and emitter location was firstly, one in which the signal representation and manipulation techniques developed in Phase I could be applied; and secondly, an area in which we could provide a demonstration of applicability within the context of the Phase I work. We found the following characteristics of the direction finding and emitter location:

- Direction finding and emitter location is an important problem area for the Army.
- Direction finding and emitter location algorithm research is an active field of research [23, 24, 25, 26, 27].
- Direction finding algorithms contain a large amount of linear algebra, i.e. matrix-vector operations, eigenanalysis, etc.
- The number of choices for computational rearrangement of linear algebra routines is significantly less than the number of choices for most signal processing operations.
- Basic signal processing operations, such as filtering and sampling rate conversion, do not play as large a role in the computations for direction finding and emitter location as they do in multirate filter banks.
- Direction finding and emitter location problems often involve multi-dimensional processing.
- Design and development of direction finding and emitter location algorithms involves modeling and understanding of physical phenomena such as multipath.
- The existing base of signal processing software contains many of the fundamental tools required for direction finding and emitter location algorithms but would require significant development to produce a complete direction finding and emitter location design and analysis system.

Based on our examinations we decided that it was not feasible to modify the Phase I work to incorporate direction finding and emitter location. It was felt that the software development required would be too large. However, we feel that the area of direction finding and emitter location is an important specialized signal processing area and

would provide a good area in which to develop an extensive system for the design and analysis of signal processing algorithms. As such, Atlantic Aerospace has proposed to develop and deliver a direction finding and emitter location algorithm testbed as the Phase II program [11].

8. Summary and Recommendations

The goal of this Phase I SBIR was to demonstrate extensions to current systems for signal representation and manipulation. The long term goal of this SBIR is to develop CAD assistants for the design, development, and test of signal processing algorithms. In Phase I we did the following:

- Surveyed several current DSP processors and developed an understanding of the common features of these processors.
- Developed an explicit representation for an abstract DSP processor that can be specialized to model specific DSP processors.
- Developed software for resource usage measurement on the abstract DSP processor.
- Interfaced the processor representation with E-SPLICE, a system for the symbolic representation and manipulation of signals, thus demonstrating hardware-specific symbolic signal manipulation.
- Rapidly extended the signal representation facilities to incorporate descriptions of multirate filter bank structures, a signal processing structure of current interest to the Army.
- Examined the area of direction finding and emitter location to determine the applicability of the signal representation and manipulation techniques to this area.
- Determined that the area of direction finding and emitter location, an important area to the Army, is an area that would benefit from application of the signal representation and manipulation techniques used in Phase I.

There are two potential directions open for future development, each with the ultimate goal of providing CAD tools for the design, development, and test of signal processing algorithms. First, the processor model developed in Phase I is limited. It does not provide a high-fidelity representation of any specific processor. A much more complete hardware modeling and analysis system could be developed as a Phase II effort. A high-fidelity model of a specific DSP processor could be written, essentially by combining a compiler and an emulator for the DSP processor of interest. This high-fidelity model could be interfaced to a signal processing algorithm description language and used to search for efficient implementations of algorithms on that specific DSP processor. This has the potential of producing a CAD system tuned to a specific processor, however, the DSP processor technology area is a rapidly evolving area and modeling today's processor may be out of date in two years. Because of the likely obsolescence of such a system, we do not recommend this approach to a Phase II

program.

The other alternative to a Phase II program is to develop a system for signal processing algorithm design and testing specialized to a specific problem area. In Phase I we demonstrated the ability of an object-oriented signal representation to allow for the rapid prototyping of signal processing algorithms for multirate filter banks. We have identified, with the help of the Army Signals Warfare Center, the area of direction finding and emitter location as a signal processing area of importance to the Army. The tools of object-oriented signal representation will allow us to develop a system for modeling of the important physical phenomena associated with this area, for applying direction finding and emitter location algorithms to either simulated or measured data, and for comparing the performance of different algorithms under the same conditions. We propose the development of this specialized signal processing testbed as the Phase II effort [11].

9. References

[1]. W. Dove, C. Myers, A. Oppenheim, R. Davis, and G. Kopec, "Knowledge Based Pitch Detection," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, Boston, MA, April, 1983, pp. 1348 - 1351.

[2]. C. Myers, A. Oppenheim, R. Davis, and W. Dove, "Knowledge Based Speech Analysis and Enhancement," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, San Diego, CA, April, 1984, pp. 39A.4.1 - 39A.4.4.

[3]. E. E. Milios and S. H. Nawab, "Interpretation-Guided Signal Processing via Protocol Analysis," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, Tampa, FL, March, 1985, pp. 42.16.1 - 42.16.4.

[4]. W. Dove, "Knowledge-Based Pitch Detection," MIT PhD Thesis, Cambridge, MA, May, 1986.

[5]. E. E. Milios, "Signal Processing and Interpretation using Multilevel Signal Abstractions," MIT PhD Thesis, Cambridge, MA, April, 1986.

[6]. G. Kopec, "The Signal Representation Language SRL," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 4, pp. 921 - 932, August, 1985.

[7]. G. Kopec, "The Integrated Signal Processing System ISP," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 4, August, 1984.

[8]. C. Myers, "Signal Representation for Symbolic and Numerical Processing," MIT PhD Thesis, Cambridge, MA, August, 1986.

[9]. C. Myers, "Symbolic Representation and Manipulation of Signals," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, Dallas, Texas, April, 1987, pp. 43.2.1 - 43.2.4.

[10]. The Mathlab Group, "MACSYMA Reference Manual," Laboratory for Computer Science, MIT, Cambridge, MA, 1983.

[11]. Atlantic Aerospace Electronics Corporation, "A Direction Finding and Emitter Location Testbed," SBIR Phase II Proposal to Army Research Office, to be submitted.

[12]. E. Lee, "Programmable DSP Architectures: Part I," *IEEE ASSP Magazine*, October 1986, pp. 4 - 19.

[13]. R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[14]. M. Vetterli, "A Theory of Multirate Filter Banks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 3, pp. 356 - 372, March, 1987.

[15]. P. Vaidyanathan, "Theory and Design of *M*-Channel Maximally Decimated Quadrature Mirror Filters with Arbitrary *M*, Having the Perfect-Reconstruction Property," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 4, pp. 476 - 492, April, 1987.

[16]. M. Vetterli, "Running FIR and IIR Filtering Using Multirate Filter Banks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-36, no. 5, pp. 730 - 738, May, 1988.

[17]. T. Nguyen and P. Vaidyanathan, "Maximally Decimated Perfect-Reconstruction FIR Filter Banks with Pairwise Mirror-Image Analysis (and Synthesis) Frequency Responses," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-36, no. 5, pp. 693 - 706, May, 1988.

[18]. M. Covell, "Representation and Manipulation of Signal Processing Knowledge and Expressions," MIT PhD Thesis Proposal, January, 1987.

[19]. H. Lee, "A Novel Procedure for Assessing the Accuracy of Hyperbolic Multilateration Systems," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 2-15, January 1975.

[20]. H. Lee, "Accuracy Limitations of Hyperbolic Multilateration Systems," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 16-29, January 1975.

[21]. H. Lee, "Accuracy of Range-Range and Range-Sum Multilateration Systems," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1346-1361, November 1975.

[22]. H. Lee and M. Wengrovitz, "Resolution Threshold of Beamspace MUSIC for Two Closely Spaced Emitters," Submitted to *IEEE Transactions of Acoustics, Speech, and Signal Processing*, November 1988.

[23]. R. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *Proceedings of RADC Spectral Estimation Workshop*, Griffiss AFBS, NY, 1979.

[24]. R. Kumaresan and D. Tufts, "Estimating the Angles of Arrival of Multiple Plane Waves," *IEEE Transactions on Aerospace and Electronic Systems*, January 1983.

[25]. G. Bienvenu and L. Kopp, "Decreasing High-Resolution Method Sensitivity by Conventional Beamformer Preprocessing," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pp. 33.2.1-33.2.4, March 1984.

[26]. R. Roy, A. Paulraj, and T. Kailath, "Direction-of-Arrival Estimation by Subspace Rotation Methods - ESPRIT," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pp. 47.2.1-47.2.4, April 1986.

[27]. M. Kaveh and A. Barabell, "The Statistical Performance of the MUSIC and Minimum-Norm Algorithms in Resolving Plane Waves in Noise," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 331-341, April 1986.

A. Processor Survey

This appendix provides a summary of some of the information collected on the single-chip DSP processors examined in our Phase I work. The reader of this section is assumed to be familiar with the basic structure of single-chip DSP processors.

A.1. TMS 32010

Chip type: TEXAS INSTRUMENTS TMS 32010 (First Generation DSP chip)

On Chip Memory

Name	RAM	ROM/EPROM	
X+ Y	288		Bytes Not partitioned
P(rogram)		3/8.0 K	Bytes

Cycles/Instruction: 1 (one operand must be in the T register for the multiply instruction); except for branch (including return and jumps) instructions which take 2.

Specific Instructions:

Instruction	Time	Details
ADD:	1 cycle	Add from RAM to accumulator
MULTIPLY:	1 cycle	Multiply 1 RAM value by value in Multiplier arg register, store result in Multiplier result register.
MULTIPLY/ ACCUMULATE	2 cycles	1-Multiply, move data to accumulator and 1-Move data to accumulator while loading a new value into Multiplier arg register

Cycle Time: Parts in this family range from 160-280 ns cycle times

Addresses Generated/Cycle: Only 1 instruction (DMOV) appears to generate addressed in parallel.

Multiply/Accumulators: 0 (load accumulator register and load T register can be combined into 1 instruction).

Inputs: 2 - T register and data bus

Intermediate precision: 32 bit

Loop overhead: Loops are not directly supported and require

1. load counter to accumulator	1 cycle
2. subtract a constant in RAM and then	1 cycle
3. check branch condition	2 cycles
4. store back to RAM.	1 cycle

For a total of 5 clock cycles/loop cycle
plus a 3 cycle startup cost of counter from program memory to RAM

Loop size: 16 bit.

Data Representation choices: 16 bit two's complement

Instruction Format: Primarily single operation per instruction.
Typical Instruction complexity: low

Width: 16 bit word
Instructions: Simple assembler type

I/O Capabilities

Number of data paths 1
Bandwidth: 1 word/2 cycles, 1 byte/cycle (mux'd port available on 320C17 but deals with 8 bit samples and compands with special hardware). No wait states available.
Number of instruction Paths: 0
Can constants be communicated from program to data: yes 3 cycles/constant
DMA capability? no
Coprocessor interface and size: (320c17) has a 16 bit interface available

Special address modes

Type: NONE
Purpose
Limits

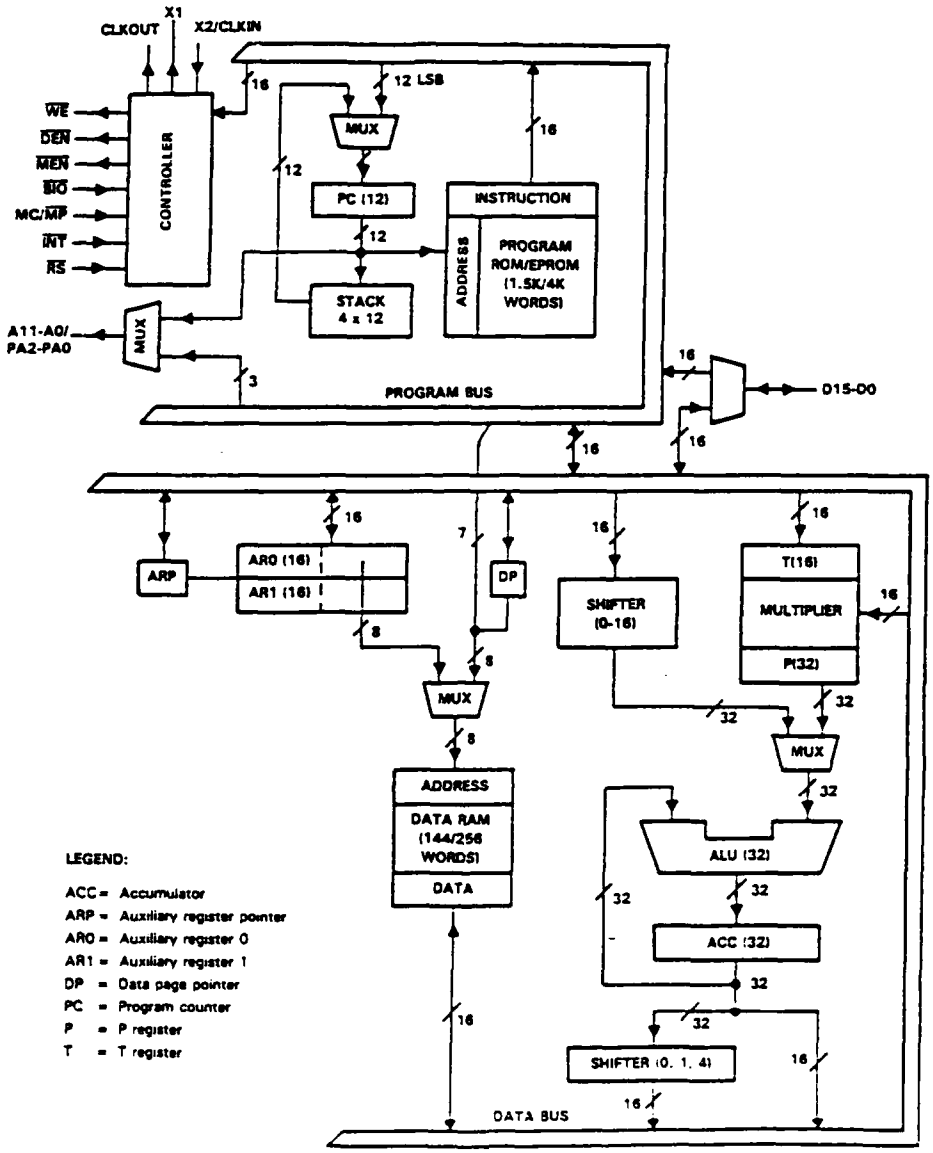


Figure 8. TMS 32010 Architecture.

A.2. TMS 32020

Chip type: TEXAS INSTRUMENTS TMS 32020 (Second Generation DSP chip)

On Chip Memory

Name	RAM	ROM/EPROM	
X+ Y	576 or 1088		Bytes Not partitioned
P(rogram)	0 or 512		Bytes

Off Chip Memory

Name	RAM	ROM/EPROM
X+ Y	64K Words	
P(rogram)	64K Words	

Cycles/Instruction: 1 (one operand must be in the T register for the multiply instruction); except for branch (including return and jumps) instructions which take 2.

Specific Instructions:

Instruction	Time	Details
ADD:	1 cycle	Add from RAM to accumulator
MULTIPLY:	1 cycle	Multiply 1 RAM value by value in Multiplier arg register, store result in Multiplier result register.
MULTIPLY/ ACCUMULATE	1 cycle	

Cycle Time: 200 ns.

Addresses Generated/Cycle: 1 a separate register ALU is provided which can calculate addresses .

Multiply/Accumulators: 0, but a 1 cycle "macro" is provided to allow repeated multiply accumulates to each be performed in a single cycle).

Inputs: 2 - T register and data bus.

Intermediate precision: 32 bit .

Loop overhead: Loops are not directly supported would require

1. load counter to accumulator	1 cycle
2. subtract a constant in RAM and then	1 cycle
3. check branch condition	2 cycles
4. store back to RAM.	1 cycle

For a total of 5 clock cycles/loop cycle
plus a 3 cycle startup cost of counter from program memory to RAM

Loop size: 16 bit

Data Representation choices: 16 bit two's complement

Instruction Format: Primarily single operation per instruction.
Typical Instruction complexity: moderate, the most complex is SQRS Square and subtract previous product.

Width: 16 bit word
Instructions: Simple assembler type

I/O Capabilities

Number of data paths 1- shared with instruction path.

Bandwidth: 1 word/2 cycles, 1 byte/cycle; wait states = $\frac{\text{memory speed - 80}}{200}$

Number of instruction Paths: 1 - shared with data path.
Can constants be communicated from program to data: yes 3 cycles/constant.
DMA capability? no.
Coprocessor interface and size: Arbitrary via bus requests.

Special address modes

Indirect addressing modes are available.

Type: NONE
Purpose
Limits

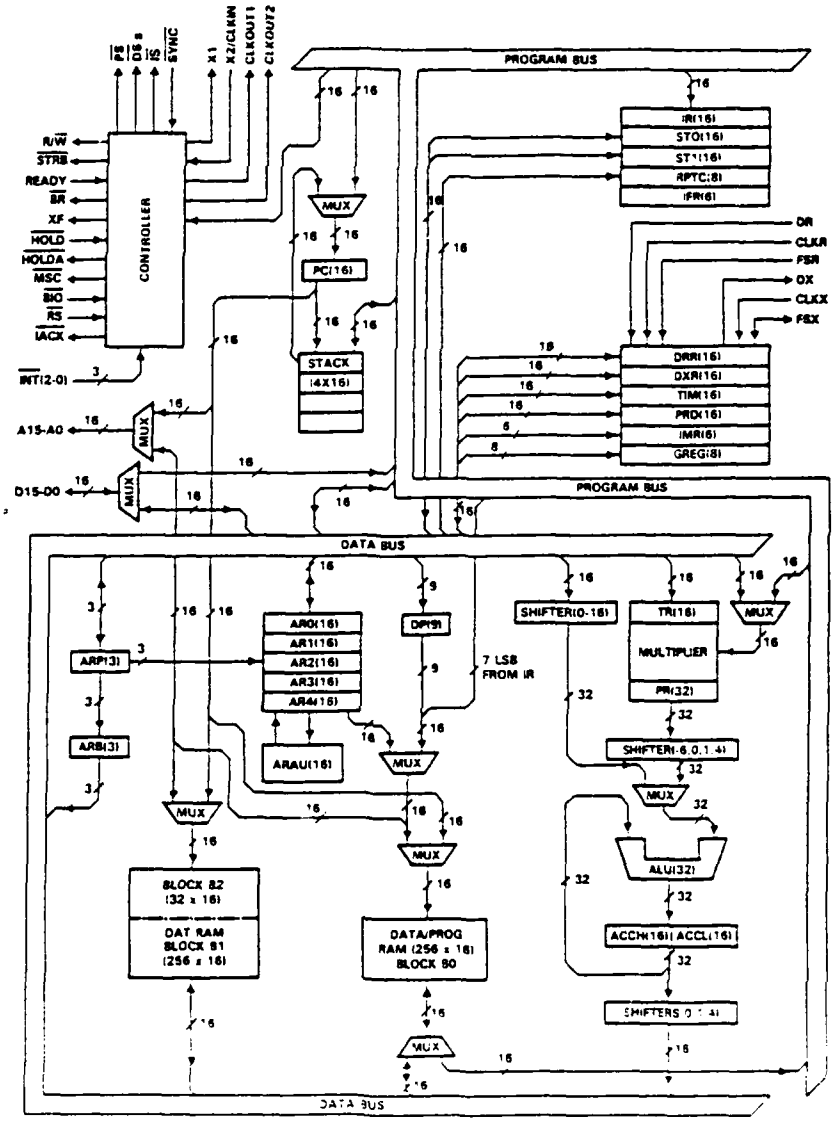


Figure 9. TMS 32020 Architecture.

A.3. TMS 32030

Chip type: TEXAS INSTRUMENTS TMS 32030 (Third Generation DSP chip)

On Chip Memory

Name	RAM	ROM/EPROM	
X+ Y	0-8K		Bytes
P(rogram)	0 or 512		Bytes

Off Chip Memory

Name	RAM	ROM/EPROM	
X+ Y	64M		Bytes

Cycles/Instruction: 1; except for branch instructions which take 4 (loop branches take 1).

Specific Instructions:

Instruction	Time	Details
ADD:	1 cycle	Add from RAM to accumulator
MULTIPLY:	1 cycle	Multiply 1 RAM value by value in Multiplier arg register, store result in Multiplier result register.
MULTIPLY/ ACCUMULATE	1 cycle	

Cycle Time: 60 ns.
Addresses Generated/Cycle: 2 a separate register ALU is provided which can calculate addresses .
Multiply/Accumulators: 1 equivalent with separate multiplier and adder.
Inputs: 3 - 2 source operands can be in memory.
Intermediate precision: 40 bit .

Loop overhead: 0

Loop size: 32 bit

Data Representation choices: 132 it floating point, 24 bit integer, 32 bit logical

Instruction Format: Some parallel instructions allowed
Typical Instruction complexity: Parallel arithmetic and store operation

Width: 32 bit word
Instructions: Assembler

I/O Capabilities

Number of data paths 1- shared with instruction path.
Bandwidth: 1 word/2 cycles, 1 byte/cycle; with 2 cycle minimum read

Number of instruction Paths: 1 - shared with data path.
Can constants be communicated from program to data: yes, no partition.
DMA capability? yes, DSP cpu not interrupted.
Coprocessor interface and size: 9 control lines.

Special address modes

Type: Circular...
Purpose: Convolution, correlation...
Limits: Register size (32 bit)...
Type: Bit Reversed...
Purpose: FFT's....
*Limits:*Register size (32 bit)

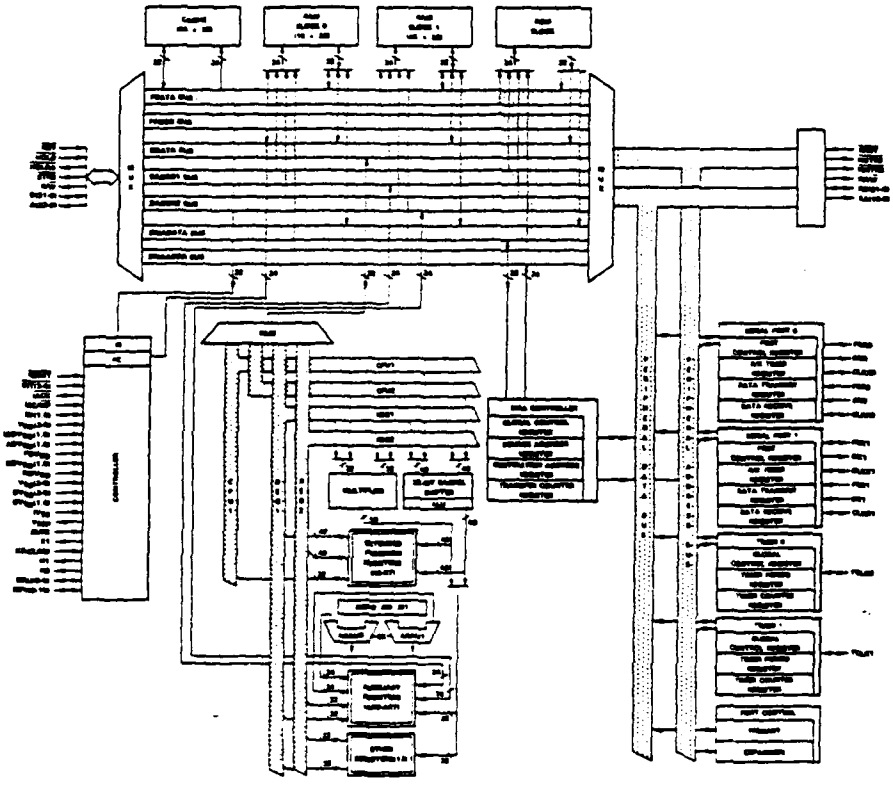


Figure 10. TMS 32030 Architecture.

A.4. AT&T DSP 32

Chip type: AT&T DSP 32

On Chip Memory On chip RAM is is two banks which should be alternately addressed for maximum speed.

Name	RAM	ROM/EPROM	
X	0-4096	0-2048	Bytes
Y	0-4096	0-2048	Bytes
P(rogram)	0-4096	0-2048	Bytes
TOTAL	4096	2048	Bytes

Off Chip Memory

Name	RAM	ROM/EPROM	
X+ Y + P	56K		Bytes

Cycles/Instruction: Generally 4, Although 1 memory access is possible/cycle.

Specific Instructions:

Instruction	Time	Details
ADD:	4 cycles	Add from RAM to accumulator.
MULTIPLY:	4 cycles	Multiply 1 RAM value by value in Multiplier arg. register, store result in Multiplier result register.
MULTIPLY/ ACCUMULATE	4 cycles	If Pipelined.
MULTIPLY/ ACCUMULATE/ STORE	4 cycles	If Pipelined.
BRANCH	4-12 cycles	Dependent upon pipeline latency.
TYPE CONVERSION	4 cycles	

Cycle Time: 40 ns (25 Mhz), 20ns (50 Mhz) for the DSP32C
Addresses Generated/Cycle: 2
Multiply/Accumulators: 1
Inputs: 5 - 4 registers and data bus.
Intermediate precision: 40 bit.

Loop overhead: 0 (CAU can execute loop logic in parallel with DAU).

Loop size: 16 bit — 16 bit integer arithmetic is all that's supported in the CAU (Control Arithmetic Unit).

Data Representation choices: 32 bit floating point, 16 bit integer, 8 bit. Floating point is 24 bit mantissa 8 bit exponent.

Instruction Format: 2 different types of instruction are available 1 for the DAU (Data Arithmetic Unit) 1 for the the CAU (Control Arithmetic Unit).
Typical Instruction complexity: Full 32 bit instruction word with pipeline control. Although both the CAU and DAU instructions each require a full 32 bit word it appears that they can be effectively overlapped since each instruction actually requires 4 cycles for execution.

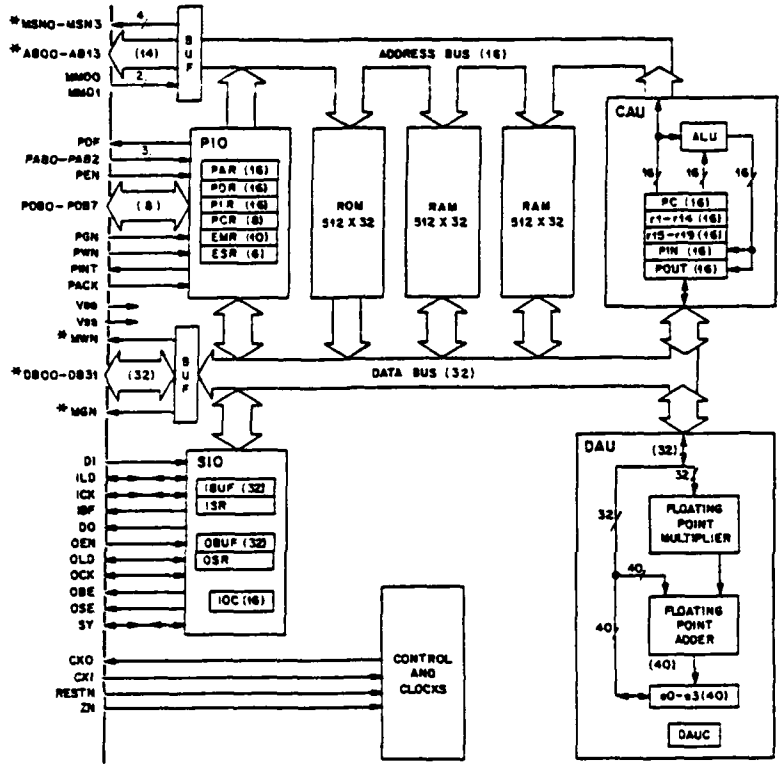
Width: 32 bit word.
Instructions: Parallel operations available.

I/O Capabilities

Number of data paths 2 - 1 parallel I/O (8 bit), 1 direct off chip I/O (32 bit)
Bandwidth: 2 bytes/21 cycles for parallel I/O (8 bit port). 4 bytes/2 cycles for off chip (32 bit port). The DSP 32C has a 16 bit parallel port rather than 8 bit, doubling the available bandwidth.
Number of instruction Paths: 1, shared with direct off chip I/O (32 bit).
Can constants be communicated from program to data: yes, no partition between program and data memory.
DMA capability? yes from CPU standpoint but steals cycles from DSP chip.
Coprocessor interface and size: ?

Special address modes

Indirect addressing modes are available.



Note: See Tables 21—27 for signal name definitions.
* *Not available on the 40-pin DIP.*

Legend:					
a0—a3	*Accumulators 0—3*	FPM	Floating-Point Multiplier	PCR	PIO Control Register
ALU	Arithmetic Logic Unit	IBUF	Input Buffer	PDR	PIO Data Register
CAU	Control Arithmetic Unit	IOC	Input/Output Control Register	PIO	Parallel I/O Unit
DAU	Data Arithmetic Unit			PIR	PIO Interrupt Register
DAUC	Data Arithmetic Unit Control Register	ISR	Input Shift Register	r1—r19	Registers 1—19
		OBUF	Output Buffer	PIN	Parallel Input Register
EMR	Error Mask Register	OSR	Output Shift Register	POUT	Parallel Output Register
ESR	*Error Source Register*	PAR	*PIO Address Register*	RAM	Read/Write Memory
FPA	Floating-Point Adder	PC	Program Counter	ROM	Read-Only Memory
				SIO	Serial I/O Unit

Figure 11. AT&T DSP 32 Architecture.

A.5. Motorola DSP 56000

Chip type: MOTOROLA DSP 56000

On Chip Memory

Name	RAM	ROM/EPROM	
X	768	768	Bytes
Y	768	768	Bytes
P(rogram)		6144	Bytes

Off Chip Memory

Name	RAM	ROM/EPROM	
X + Y	384K		Bytes
P	192K		Bytes

Cycles/Instruction: Generally 2.

Specific Instructions:

Instruction	Time	Details
ADD:	2 cycles	Add from RAM to accumulator
MULTIPLY:	2 cycles	Multiply 1 RAM value by value in Multiplier arg register, store result in Multiplier result register.
MULTIPLY/ ACCUMULATE	2 cycles	If Pipelined (2n + 9).
BRANCH	4-6 cycles	Dependent upon complexity of test.

Cycle Time: 50 ns (20 Mhz)
Addresses Generated/Cycle: 2-Separate address ALU.
Multiply/Accumulators: 1
Inputs: 3.
Intermediate precision: 40 bit .

Loop overhead: 3 cycle startup.

Loop size: 16 bit (Address ALU limit)

Data Representation choices: Words (24 bit), long words (48 bit)

Instruction Format: Single operation per instruction.
Typical Instruction complexity: Moderate.

Width: 24 bit.
Instructions: Assembler.

I/O Capabilities

Number of data paths 1 24 bit data shared with program, 1 parallel I/O.
Bandwidth: 3 bytes/cycle (1 access per cycle).
Number of instruction Paths: 1, shared with direct off chip I/O (32 bit).
Can constants be communicated from program to data: yes, 2 + number of moves.
DMA capability? yes from host standpoint.
Coprocessor interface and size: 8 Bit parallel.
Addresses Busses: 1 3-way multiplexed.

Special address modes

Indirect addressing modes are available.

Type: Reverse Carry.
Purpose: FFT address generation.
Limits: 65536 point fft.

Type: Modulo modification.
Purpose: Circular shifts, etc.
Limits: mods from 2-32768.

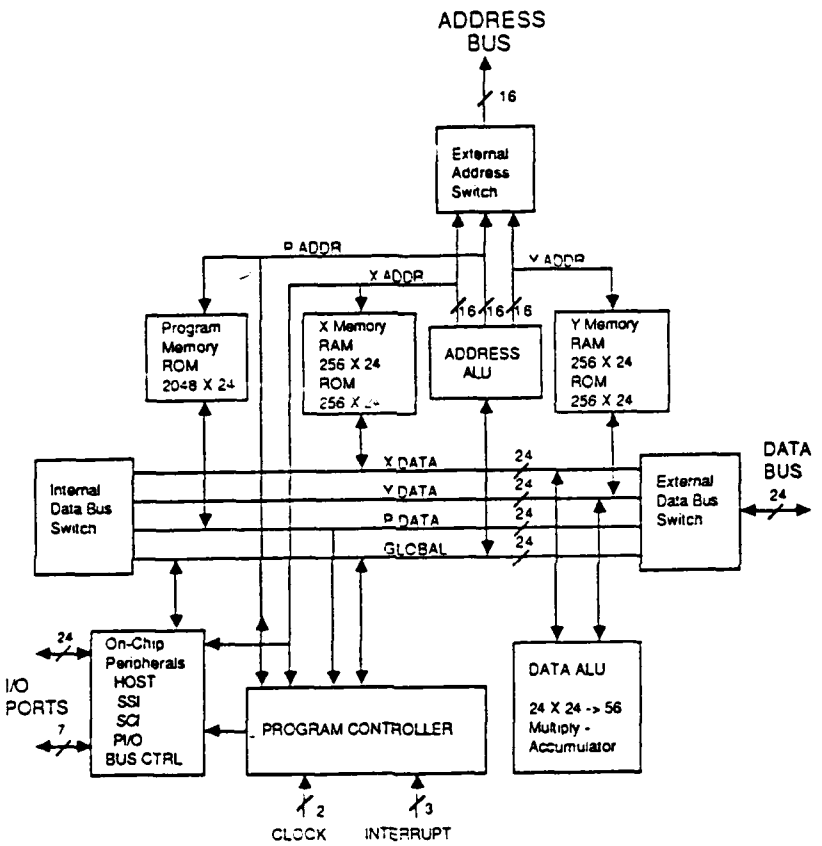


Figure 12. Motorola DSP 56000 Architecture.